Post-OCR error correction : a hybrid approach to solve the post-OCR correction problem

Davide Raimondi

Corresponding author(s). E-mail(s): davide.raimondi2@studenti.unimi.it;

Abstract

This brief article was written for the exam of information retrival 2022/2023 at University of Milan to show and analyze the performance of a novel approach to the post-OCR error correction problem. OCR has been an active area of research for a long time but the result of OCR systems often contains errors, particularly when the text is not well written or well preserved. To correct the output of an OCR system, it is necessary to address three main challenges: word segmentation, error correction and error detection. The proposed approach aims to create a unique system that uses different techniques to solve these problems. The dataset used for this task is the same one used for the ALTA 2017 Challenge.

1 Introduction

The post-OCR problem has been widely studied in the last ten years and there are numerous papers on this topic. In [1], three different approaches to post-OCR correction are explained; the human approach, the lexical approach and the statistical approach. The human approach uses volunteers to correct the errors, meaning a very precise result but not very efficient or scalable process. The idea behind the lexical approach is to find and correct the errors produced by an OCR system using the distance from the word in a lexicon. The limits of this approach are basically three: it's challenging to correct segmentation errors, a comprehensive lexicon is needed and the "real word error", for example "hell. my name is Jane", can't be corrected. The prevailing approach is the statistical one, in which the solution involves the development of a statistical model . To solve the segmentation in [2] a NMT (neural machine translator) at char level was used with good results. In [1] the identification and correction of the incorrect word are discussed. They proposed to use a pre-trained named entity recognition (NER) model (BERT, [3]) for the identification and a seq2seq Transformers for the correction. This approach is also validated by [4] which shows some of the

techniques and the results from a competition on post OCR correction. The model proposed in this work is composed by a sequence-to-sequence Transformers that, given a text without spaces, inserts them in the correct spot to solve the segmentation problem. Subsequently a token classification model is used to classify each word as correct or wrong. Finally a novel approach is used that combines the previous model, a lexicographical correction solution and some ideas from evolutionary algorithm to correct the documents. Given the unsatisfactory results, analysis are conducted on the dataset and on the results themselves to explain why the approach doesn't work in the considered context.

2 Dataset

For the task the dataset proposed for Alta's 2017 challenge [5] is used. This is composed of 6000 documents produced from a state-of-the-art OCR system, with corrections provided by some volunteers. Starting from this, the dataset is divided in two parts: the first to train and validate the performance (70%) and the other to test the whole model.

3 Model

The proposed model is built from three elements: A seq2seq Transformer that works as an NMT that translates a text without spaces to text with spaces in the correct spot, a fine-tuned BERT that tries to solve the error detection problem assigning a label to each word based on the correctness of the word, and a lexicographical algorithm that tries to use some statistical knowledge about the errors in OCR's output and the knowledge of the context to substitute the wrong words with similar words, in terms of edit distance, in the lexicon.

3.1 custom Transformer for segmentation

The correction method that is proposed has the problem that it can't handle the wrong segmentation problem. To avoid this, the work proposes to train and use a seq2seq NMT model only for the task of separating words, and then use this as a pre-processing step. The output of the text passed to this model can become the input of the model that is used to detect and correct post-OCR errors. The problem with this idea is that the success of the subsequent models depends on the correctness of the segmentator. To build this first block, the same pre-processing step as in [2] is used. In summary, the pre-processing of the input removes the spaces from the text, replaces each character with a number, and inserts some special symbols to signal the start and end of a sequence. There is also the problem of the variable length of the documents. A maximum number of characters is chosen, and the documents that are longer than the maximum length are split on a space using the most central space. When a document is too long after a split, the procedure is repeated. When the length of all documents is less than the maximum length, all the sequences are padded with a special symbol. Different encodings are tested to build the embedding for the model, and in the end the Pytorch default embedding is used. In this is encoded

either the character and its position, which is necessary to apply attention. One of the main advantages of this approach is that it can be applied to any English text. This is true under the assumption that, in the task of producing a document with spaces, a document without them is similar to a text with post-OCR's errors. Due to the scarcity of computational power, however, only the output of the training set (4200 documents) is used. To build the model itself, PyTorch is used (in particular, the nn.Module and nn.Transformer classes).

3.2 BERT for error detection

Given the result of [1] using BERT to detect words with errors appears to be a promising approach. The first thing to do is to associate to each word in the input documents its label using the ground truth documents. For each word in the input document, a window of words in the ground truth is considered, starting from the word's position . If the input word is within the window, a positive label is assigned to that word, indicating that the word is correct; otherwise, a negative label is assigned. Given the types of segmentation errors [6] the difference in length of the input and ground truth documents is used to choose the window's size. In some cases, using this technique may result in a few false positives, but the likelihood of this occurrence isn't very high. Next the words are decomposed using sub-tokens (following the encoding used in BERT, which is based on the WordPieces algorithm). The label of a word is paired with the first token, while the others are assigned a default value (-100). When the word is reconstructed to check its correctness, the minimum value of the sub-token that compose the word is used. The model employed is distilBERT [7], chosen due to limited computational resources. The hyperparameters are selected empirically, drawing inspiration from values proposed for similar models that perform NER. The pre-processing and the fine-tuning step were carried out using Hugging Face models and libraries.

3.3 hybrid algorithm for the word correction

Considering the limitations of a purely lexical approach, this project tried a method for attempting lexical correction based on the context of the words within the document. The criterion to find similar words in the lexicon is based on the concept of Levenshtein distance. Initially BERT is used to perform error detection on a document, producing a list of incorrect words. Starting from the first incorrect word, some regex patterns are created to identify the words within a specified Levenshtein distance in the lexicon. Each regex pattern is paired to a specific type of error. For instance, to simulate a substitution error a regex pattern is generated, replacing the letter in position "i" in the word with the "
w" symbol. This pattern matches all words in the lexicon where the other letters are identical to the starting words and there is any lowercase letter in position "i". Each word matched with a pattern is assigned a score based on the probability of having the type of errors from which the pattern is generated, as described in [6]. Using two parameters (number of maximum words considered and maximum Levenshtein distance), the algorithm retrieves all the words in the lexicon at a specified Levenshtein distance, starting from 1. Then, if the number of words is less than the number of

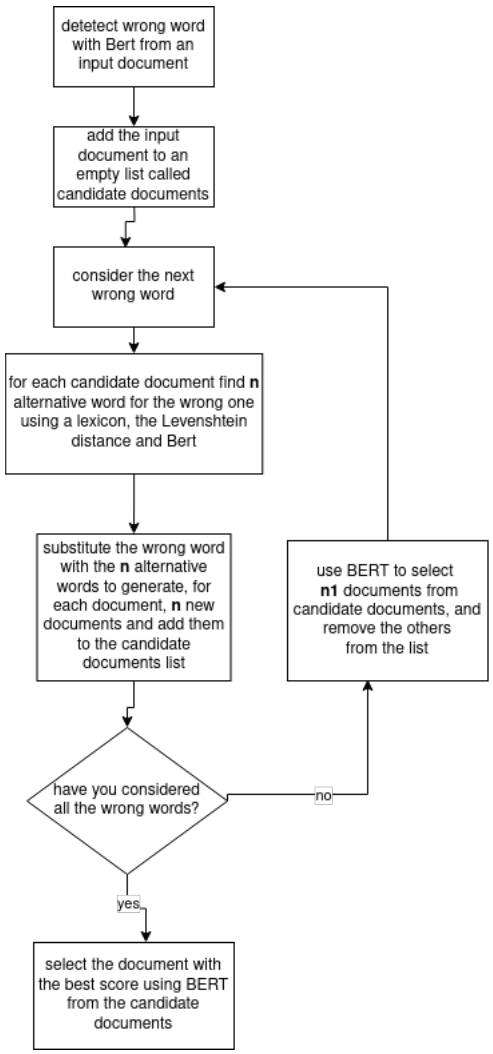

Fig. 1: flow-chart of errors correction's module

maximum word and if the Lenvenshtein distance is less than the maximum distance, the Lenvenshtein distance is increased, and the process is repeated. For each word found with this approach, a new document is created by replacing the original word with the found one. Now, BERT can be used to evaluate the correctness of the new word in the context; this step allows a lexical approach that considers the context. The score given by BERT for the selected words in the new document and the error score associated with the regex pattern are employed to select a specified number of new documents. The algorithm that makes this selection is inspired by the roulette wheel selection used in some evolutionary algorithms. A custom distribution is created based on the scores of the documents, and a finite number of them is extracted. The

epoch	avg cross-entropy loss
1	1.22
2	0.24
3	0.15
4	0.13
5	0.11

Table 1: average cross-entropy loss on the train set for epoch

number of documents to extract is another important parameter of the model. Now the algorithm has produced some alternative documents. For each of them, another word that is labeled as incorrect in the original document is considered, and the error correction sequence is repeated. Given that the second time the correction is repeated the steps are executed for all the generated documents, the number of documents will soon become exponential. To avoid this occurrence, after each correction step, another selection of the candidate documents is carried out. There is another parameter that determines the maximum number of documents to be retained, and then another roulette wheel selection is performed. This time, to select the more promising documents, the sum of all the scores of the words in the documents given by BERT is considered. The idea behind this approach is to try to do a lexical correction, using BERT for the correction and employing the evolutionary approach to retain the most promising documents. This allows to explore the space of possible solutions without considering an exponential number of documents. The diagram here 1 summarizes the functioning of the correction algorithm. One of the main advantages of this type of algorithm is that the correction step is highly parallelizable, and it seems possible to realize an implementation that runs entirely on a GPU.

4 Results

Given the three blocks that compose the model, some evaluations are performed on each of these, followed by a global evaluation of the entire system.

4.1 Results of the Transformer for segmentation

In this section, the results of the Transformer in terms of cross-entropy loss are presented. The table 1 contains the cross-entropy loss on the train dataset, while the table 2 lists the hyperparameters used for the Transformers. The table 3 contains the results on the test set and shows the best results obtained in terms of the cross entropy loss and the Levenshtein distance ratio, calculated as 1 - normalized Levenshtein distance.

4.2 AdvDiffusion experiments

The results of the Transformer are unsatisfactory, rendering the model unsuitable for the project. Consequently, the detection and correction of the words are done on the original documents. To enhance the model's performance, it is necessary to either train

hyperparameter	value
number of epochs	5
number of attention heads	8
number of encoding layers	3
number of decoding layers	3
dimension of feed-forward layer	1024

Table 2: Transformer hyperparameters

avg cross-entropy loss	avg Levenshtein distance ratio
6.90	0.29

Table 3: results on test set

the Transformer on a greater amount of data or to find a pre-trained model with a char-based encoding to fine-tune. Given the pre-processing step, any English document can be used to train the segmentator, solving the problem of finding labeled data. Another improvement to be implemented is the insertion of an early stopping policy to prevent overfitting. Fine-tuning the model's hyperparameters is an important step to enhance its performance. While adjusting the learning rate is the only change made to enable the model's functionality, building a model with good performance requires systematic testing of the other hyperparameters. With the exception of the early stopping policy, other proposals suffer limitations due to a lack of computational power (the model is trained without a GPU). Consequently, training time has become extensive, with about one day of training with the current dataset, a very small number of epochs (5), and without testing hyperparameters. In the current scenario, it is possible to add an early stopping policy to the train, but the model isn't good enough to be used, even on the train dataset, on which the model is overfitted. Thus, early stopping might prevent overfitting, but this doesn't ensure satisfactory model performance.

4.3 results of the error detection module

In this section, the results obtained using BERT for error detection are briefly discussed. The task involves token classification, so the metrics used are the same as for a two-class classification problem. Accuracy, precision, recall, and f1-score are reported in 4. All the metrics are calculated individually for each document and the average along with the std deviation is reported, considering all the documents in the test dataset. In image 2 is plotted the confusion matrix, while in this table 5 are inserted the hyperparameters used for the fine-tuning.

The results of BERT are generally positive; however, the classes are highly unbalanced, which leads to an important number of false positives. In the context in which the system is inserted, a false positive is preferable to a false negative. This is because if a correct word is labeled as negative, changing it will impact the document's correctness. On the other hand, ignoring a wrong word doesn't reduce the correctness of the document. Therefore, utilizing this module as an error detector has the potential to improve the original text. To further refine the model, a systematic selection

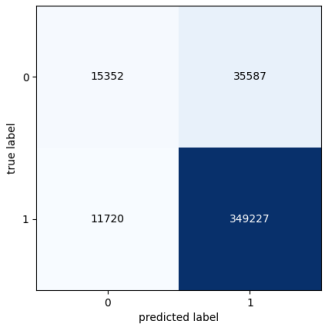

Fig. 2: Confusion matrix

metrics	avg	std. deviation
accuracy	0.88	0.13
precision	0.90	0.14
recall	0.96	0.07
fi-score	0.92	0.11

Table 4: metrics of fine-tuned error detector

metrics	values
learning rate	2e-6
weight decay	0.01
number of train epochs	5

Table 5: hyperparameters of BERT fine-tuning

of the hyperparameters is necessary, and potentially, a larger dataset can improve the performance.

4.4 result of error correction module

The correction model involves numerous parameters that influence both the performance and the computational power required to make the corrections. For the experiments, the maximum Levenshtein is set at two, a maximum of twenty-five possible words for each incorrect one is considered, and three possible documents to conserve after each correction step are selected. To evaluate the results of the error's correction module, only a small sample of the complete dataset (50 documents) is considered due to the extensive execution time of the module. The results are expressed in terms of the Levenshtein ratio. In this table 6, the Levenshtein distance ratio between the ground truth and the unmodified dataset (on the left) and the dataset passed to the correction module (on the right) are listed . While the actual correction module, if applied to a bad segmentation word, can only increase the Levenshtein distance (as

distance of raw input	distance of corrected input
0.97±0.04	0.88 ±0.04

Table 6: results of the correction module

it tries to substitute two words with one existent word), according to [6] and some naive analysis done on the length difference of the documents, the number of segmentation errors is, on average, too low to justify the results. After some analysis of the dataset, two key points have emerged. The first one is that the types of errors in the datasets do not seem to align with those reported in [6]. Upon a superficial examination, it's evident that some of the documents have numerous words that are completely incorrect, with a Levenshtein distance slightly greater than one or two. The correction system struggles to find the correct alternative for these words, and trying to correct them can improve the distance from the ground truth. The cause may be the different timestamp of the two papers: the [6] paper was written in 2019, while the dataset used in ALTA [4] dates back before 2017. The second point is based on the ground truth that, from a superficial examination, seems to contain many incorrect words. To confirm this, the number of words in the datasets that aren't in the lexicon (NLTK English dictionary) has been checked. From this analysis, it appears that 46% of the total words in the ground truth aren't in the lexicon. This implies that if one of the correspondent words in the input is labeled as an error, correcting it using this module can only improve the Levenshtein distance. Given this second point, a lexicon approach on this dataset isn't feasible, at least with this lexicon.

5 Conclusion

In the end, this work doesn't provide good enough results to become a starting point for improvement in this field. A Transformer trained with a large dataset that, given a document with errors, tries to produce a sequence without them may be a better solution to solve the entire problem. At least this work confirms that a fine-tuned version of BERT (also distilBERT) can be used to perform error detection in postOCR's system. It also taught me, in a practical way, the importance of conducting an accurate analysis of the dataset before initiating the modeling process. In future works, it's possible to try the approach on different datasets and see if the detection and correction parts perform better. If this proves to be successful, a slight improvement can be made parallelizing the correction part to make it feasible in terms of time execution. Another enhancement in the algorithm could involve checking if multiple subsequent words are incorrect and attempting to merge them to address part of the segmentation problem.

References

[1] Nguyen, Thi Tuyet Hai and Jatowt, Adam and Nguyen, Nhu-Van and Coustaty, Mickael and Doucet, Antoine, (Association for Computing Machinery, New York, NY, USA, 2020), JCDL '20, p. 333–336. https://doi.org/{10.1145/3383583.3398605}. URL https://doi.org/10.1145/3383583.3398605

[2] Nastase, Vivi and Hitschler, Julian, in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC) 2018)*

[3] Devlin, Jacob and Chang, Ming-Wei and Lee, Kenton and Toutanova, Kristina, Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)

[4] Chiron, Guillaume and Doucet, Antoine and Coustaty, Mickaël and Moreux, Jean-Philippe, in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 1 (IEEE, 2017), pp. 1423–1428

[5] Molla, Diego and Cassidy, Steve, in *Proceedings of the Australasian Language Technology Association Workshop 2017* (2017), pp. 115–118

[6] T.T.H. Nguyen, A. Jatowt, M. Coustaty, N.V. Nguyen, A. Doucet, *Deep Statistical Analysis of OCR Errors for Effective Post-OCR Processing.* https://doi.org/10.1109/JCDL.2019.00015. Pages: 38

[7] Victor Sanh and Lysandre Debut and Julien Chaumond and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter (2020)